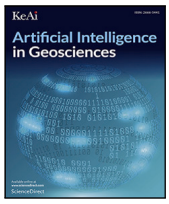
Artificial Intelligence in Geosciences 6 (2025) 100146

Contents lists available at ScienceDirect

Artificial Intelligence in Geosciences

journal homepage: www.keaipublishing.com/en/journals/artificial-intelligence-in-geosciences



Original research articles

Online learning to accelerate nonlinear PDE solvers: Applied to multiphase porous media flow

Vinicius L.S. Silva [a,b,c,*], Pablo Salinas [e], Claire E. Heaney [a,d], Matthew D. Jackson [b], Christopher C. Pain [a,d]

[a] Applied Modelling & Computation Group, Imperial College London, UK
[b] Novel Reservoir Modelling and Simulation Group, Imperial College London, UK
[c] Petroleo Brasileiro S.A. (Petrobras), Rio de Janeiro, Brazil
[d] Centre for AI Physics Modelling, Imperial-X, Imperial College London, UK
[e] OpenGoSim, Leicester, UK

ARTICLE INFO

Dataset link: https://github.com/viluiz/solver

Keywords:
Nonlinear PDE solver
Machine learning
Online learning
Numerical relaxation
Multiphase flows
Porous media

ABSTRACT

We propose a novel type of nonlinear solver acceleration for systems of nonlinear partial differential equations (PDEs) that is based on online/adaptive learning. It is applied in the context of multiphase flow in porous media. The proposed method rely on four pillars: (i) dimensionless numbers as input parameters for the machine learning model, (ii) simplified numerical model (two-dimensional) for the offline training, (iii) dynamic control of a nonlinear solver tuning parameter (numerical relaxation), (iv) and online learning for real-time improvement of the machine learning model. This strategy decreases the number of nonlinear iterations by dynamically modifying a single global parameter, the relaxation factor, and by adaptively learning the attributes of each numerical model on-the-run. Furthermore, this work performs a sensitivity study in the dimensionless parameters (machine learning features), assess the efficacy of various machine learning models, demonstrate a decrease in nonlinear iterations using our method in more intricate, realistic three-dimensional models, and fully couple a machine learning model into an open-source multiphase flow simulator achieving up to 85% reduction in computational time.

1. Introduction

The numerical solution of partial differential equations (PDEs) is an ubiquitous tool for modelling physical phenomena. Among PDE problems, multiphase flow in porous media is crucial for understanding, developing and managing subsurface reservoirs, with applications in areas such as carbon sequestration, hydrocarbon recovery, geothermal energy extraction, and groundwater resources. However, the intricate coupling between the governing equations and their inherent nonlinearity present substantial challenges to the numerical solution (Aziz, 1979; Jackson et al., 2015; Li and Tchelepi, 2015; Pour et al., 2023). Traditionally, Newton methods in monolithic fully-implicit formulations have been employed to solve these discretized nonlinear equations; however, sequential methods are emerging as a competitive alternative due to their flexibility and extensibility. Sequential methods allow the use of specialized solvers for the solution of each equation in multiphysics problems (Salinas et al., 2017a; Jiang and Tchelepi, 2019; Freitas et al., 2020). The Picard iterative solver can function as a sequential method, which can be conceptualized as a nonlinear block Gauss–Seidel problem or a sequential fixed-point iteration (Silva et al., 2021). Compared to Newton-like methods, Picard iterative solvers have less stringent convergence criteria (Elman et al., 2005; Lott et al., 2012). Newton methods can also be used in sequential approaches within both inner and outer loops. Their quadratic convergence makes them generally faster than Picard iterations (Ortega and Rheinboldt, 1970; Jenny et al., 2006; Wong et al., 2019).

Significant research has been performed to accelerate sequential methods. To this end, the use of numerical relaxation (Press et al., 2007) has shown promising results. Salinas et al. (2017a) introduced an acceleration technique employing numerical relaxation, targeting the saturation field in multiphase porous media problems. Jiang and Tchelepi (2019) implemented three distinct nonlinear acceleration techniques in the sequential-implicit fixed point method, with numerical relaxation outperforming both quasi-Newton and Anderson acceleration.

The widespread success of machine learning across various domains has recently motivated its application in accelerating the convergence

* Correspondence to: Royal School of Mines, Imperial College London, South Kensington Campus, London SW7 2AZ, UK.
E-mail address: v.santos-silva19@alumni.imperial.ac.uk (V.L.S. Silva).

https://doi.org/10.1016/j.aiig.2025.100146
Received 25 April 2025; Received in revised form 19 June 2025; Accepted 12 July 2025
Available online 23 July 2025
2666-5441/© 2025 The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

of numerical PDE simulations. Greenfeld et al. (2019) proposed a new framework for multigrid PDE solvers, where a single mapping from discretization matrices to prolongation operators is learned using a neural network in an unsupervised learning procedure. Hsieh et al. (2019) proposed an approach to learn a fast iterative PDE solver tailored to a specific domain. It achieved significant speedups compared to standard formulations; however, it only works for linear solvers. Oladokun et al. (2020) implemented a random forest regression to generate the linear convergence tolerance in a nonlinear solver, resulting in a reduction of linear iterations. Kadeethum et al. (2022) used the prediction from a machine learning-based reduced-order model as a initial guess for the nonlinear solver iterations. Their approach maintains the full order model accuracy while accelerating the nonlinear solver convergence. In a previous work (Silva et al., 2021), we proposed a machine learning acceleration to determine the value of the numerical relaxation. We used a random forest model (trained offline) and a selection of dimensionless parameters for the training and controlling the machine learning algorithm. The machine learning acceleration was able to generalize, and improve the stability and efficiency of the nonlinear solver.

A promising approach in the field of machine learning comes in the form of learning from online/evolving data streams (Read et al., 2012; Kirkpatrick et al., 2017; Gomes et al., 2017a; Chen and Liu, 2018; Montiel et al., 2020; Hoi et al., 2021). It can provide an attractive alternative to accelerate nonlinear PDE solvers, since in each numerical simulation several nonlinear iterations are performed, and they can be used to improve the convergence of the nonlinear solver. We can highlight two main branches of algorithms to train the machine learning model, instance-incremental methods (Cauwenberghs and Poggio, 2000; Oza and Russell, 2001; Losing et al., 2016; Hoi et al., 2021) and batch-incremental methods (Breiman, 1999; Polikar et al., 2001; Wang et al., 2003; Montiel et al., 2020). Instance-incremental learning uses only one sample to update the machine learning model at each time, where batch-incremental learning uses a batch of multiple samples to update the model each time. Read et al. (2012) compared these two approaches for the task of classification of evolving data streams. They showed that both methods perform similarly given a limited resource, and that the optimal batch size depends on the problem in consideration.

In this work, as a starting point, we perform a sensitivity in the machine learning models and features of the machine learning acceleration proposed in our previews work (Silva et al., 2021). Then we use an online learning approach to improve/update the machine learning model in run time, with the aim of continuously adapting to changes in the numerical PDE simulation. To the best of the authors' knowledge, this is the first work to use online learning to accelerate a numerical PDE solver.

We list the contributions of our work as follows:

- We propose an online/adaptive learning acceleration for nonlinear PDE solvers, and apply it to a broad class of multiphase porous media problems. It is able to control the convergence of the nonlinear solver by learning from previous nonlinear iterations of the running numerical simulation.
- We investigate the best set of features (dimensionless numbers) used as input parameters for the proposed acceleration and compare different machine learning models for controlling the convergence of the solver.
- We fully integrate a machine learning model into the nonlinear solver of an open-source multiphase flow simulator, achieving up to 85% reduction in computational time (only using the offline training, the online update would further improve the results).
- The proposed methodology can be applied to other types of nonlinear solvers provided they use a numerical relaxation or another tuning parameters that influences the solver convergence.

The remaining of this paper is structured as follows. We present the governing equations of the multiphase porous media flow and the Picard iterative solver in Section 2. Section 3 describes the proposed adaptive learning acceleration. Following that, the test cases are detailed in Section 4 and the results are presented in Section 5. Finally, a discussion and concluding remarks are provided in Sections 6 and 7, respectively.

2. Background and problem setting

In this section, we show the governing equations of the multiphase porous media flow problem. Then we describe the Picard iterative solver and the numerical relaxation used here.

2.1. Governing equations and discretization

We report the governing equations for incompressible porous-media flow considering capillary pressure and gravity. The mass-balance equation is

$$\phi \frac{\partial S_\alpha}{\partial t} + \nabla \cdot \mathbf{u}_{\boldsymbol{\alpha}} = Q_\alpha, \tag{1}$$

where α represents an immiscible-fluid phase and t is time. ϕ is the rock porosity, $\mathbf{u}$ is the Darcy velocity, S is the saturation, and Q_α is a source term.

The relation between pressure and the Darcy velocity is given by the multiphase Darcy's law for phase α as

$$\mathbf{u}_{\boldsymbol{\alpha}} = \frac{k_{r\alpha}\mathbf{K}}{\mu_\alpha}(-\nabla p_\alpha + \rho_\alpha g \nabla z), \tag{2}$$

where μ_α, $k_{r\alpha}$, and $\mathbf{K}$ are the viscosity, relative permeability, and permeability tensor, respectively. ρ is the density, p is the pressure, g is the gravitational acceleration, and ∇z is the gravity direction.

Considering a wetting (w) and non-wetting (nw) phase and including capillary pressure, p_c, the system of equations is completed by the constraints

$$p_c = p_{nw} - p_w, \tag{3}$$

$$S_w + S_{nw} = 1. \tag{4}$$

Our approach adopts a spatial discretization strategy using the double control volume finite element method, as elaborated in Salinas et al. (2017b). This method offers an enhancement over the traditional control volume finite element method, particularly for highly distorted meshes, as discussed in Fung et al. (1992), Durlofsky (1993), Jackson et al. (2015) and Gomes et al. (2017b). In this framework, velocity is discretized via finite elements, while pressure and saturation are handled through control volumes. For time discretization, we employ a θ-method. Here, θ is adaptively adjusted between 0.5 (Crank–Nicolson scheme) and 1 (equivalent to implicit Euler), guided by the total variation diminishing criterion (Pavlidis et al., 2014). The solution of the equations reported here are implemented in the open-source code IC-FERST (Imperial College Finite Element Reservoir SimulaTor) (Gomes et al., 2017b; Salinas et al., 2017a,b; Obeysekara et al., 2021). Further details about IC-FERST can be found in https://multifluids.github.io/.

2.2. Nonlinear solver

We use a sequential method (Picard iterative method) to solve the discretized nonlinear system of equations, constituted by Eqs. (1), (2), (3), and (4). The method, outlined in Salinas et al. (2017a), is depicted in Fig. 1 and involves three iterative loops. The time loop is indicated by a solid line in Fig. 1, the coupling between saturation and pressure (nonlinear outer loop) by a dotted line, and the iteration over saturation and velocity (nonlinear inner loop) by a dashed line. Within the inner loop, saturation is calculated first, followed by an update in velocity based on the newly computed saturation. This cycle

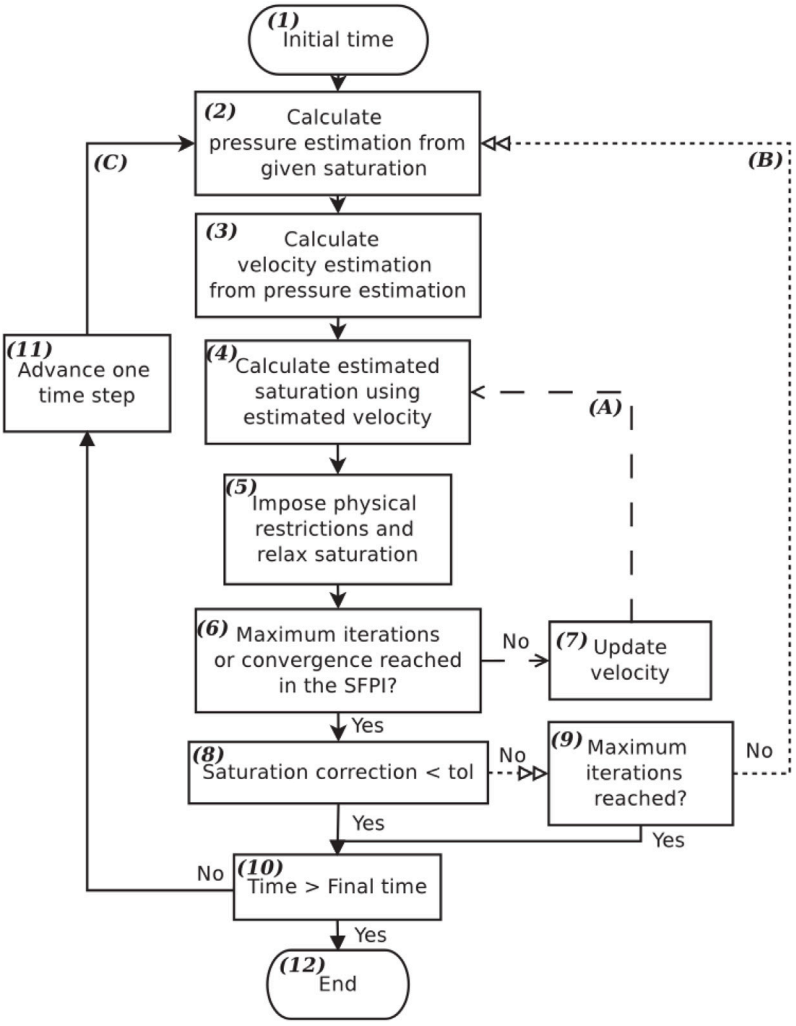


Fig. 1. Sequential nonlinear solver loops.
Source: Reprinted from Salinas et al. (2017a).

repeats until either convergence is achieved or the maximum number of iterations is reached. Subsequently, in the outer loop, pressure is recalculated based on the updated saturation, which in turn leads to a new velocity calculation derived from the pressure estimate. This initiates a new iteration in the inner loop. The entire process continues until either convergence is achieved or the maximum iteration limit is met.

To enhance the convergence rate of the nonlinear solver, a relaxation parameter is implemented (Salinas et al., 2017a; Jiang and Tchelepi, 2019; Silva et al., 2021). This technique updates the saturation by integrating the newly computed field with saturations from prior inner loop iterations. The calculation for the new saturation is as follows

$$S^k = \omega^k \tilde{S}^k + (1-\omega^k)S^{k-1} + (1-\omega^k)^{\beta+1}\omega^k(S^{k-2} - S^{k-1}), \tag{5}$$

where S^k is the updated saturation, which is calculated using the saturation $\tilde{S}^k$ from step (4) in Fig. 1. k denotes the current iteration of the inner loop. The relaxation parameter is represented by ω, and β is the exponent that determines the relative significance of S^{k-1} and S^{k-2}. Here, we adopt a constant value of 0.4 for β, following the methodology outlined in Salinas et al. (2017a).

In each outer nonlinear iteration, the initial relaxation parameter w^0 sets the stage for calculating subsequent relaxations w^k. The latter are computed to optimize the convergence rate by minimizing the residual saturation (Salinas et al., 2017a). A critical consideration is the selection of w^0 for each outer nonlinear iteration. While a simple approach might use a constant value across all iterations, the relaxation parameters must be carefully balanced: small enough to prevent divergence, yet large enough to expedite convergence. The choice of w^0 significantly influences the convergence of the nonlinear loops, but its optimal value is not predetermined and varies with each specific problem. Furthermore, even the most effective static value may lead to a higher number of iterations compared to a well-adjusted dynamic relaxation factor (Küttler and Wall, 2008; Jiang and Tchelepi, 2019; Silva et al., 2021).

3. Online learning acceleration

The relaxation parameter is of paramount importance to accelerate the nonlinear solver convergence. Therefore, we propose a novel approach to adjust the relaxation factor in each outer nonlinear iteration, where we adapt the machine learning model to the changes in the numerical simulation during run time, as shown in Fig. 2 and Algorithm 1. We start with an offline part, where the machine model is trained in a dataset generated by a simple two-dimensional reservoir model. The machine learning model is designed to take simulation properties, dimensionless numbers, and the number of inner nonlinear iterations at each outer nonlinear iteration as inputs. Its output is the calculated

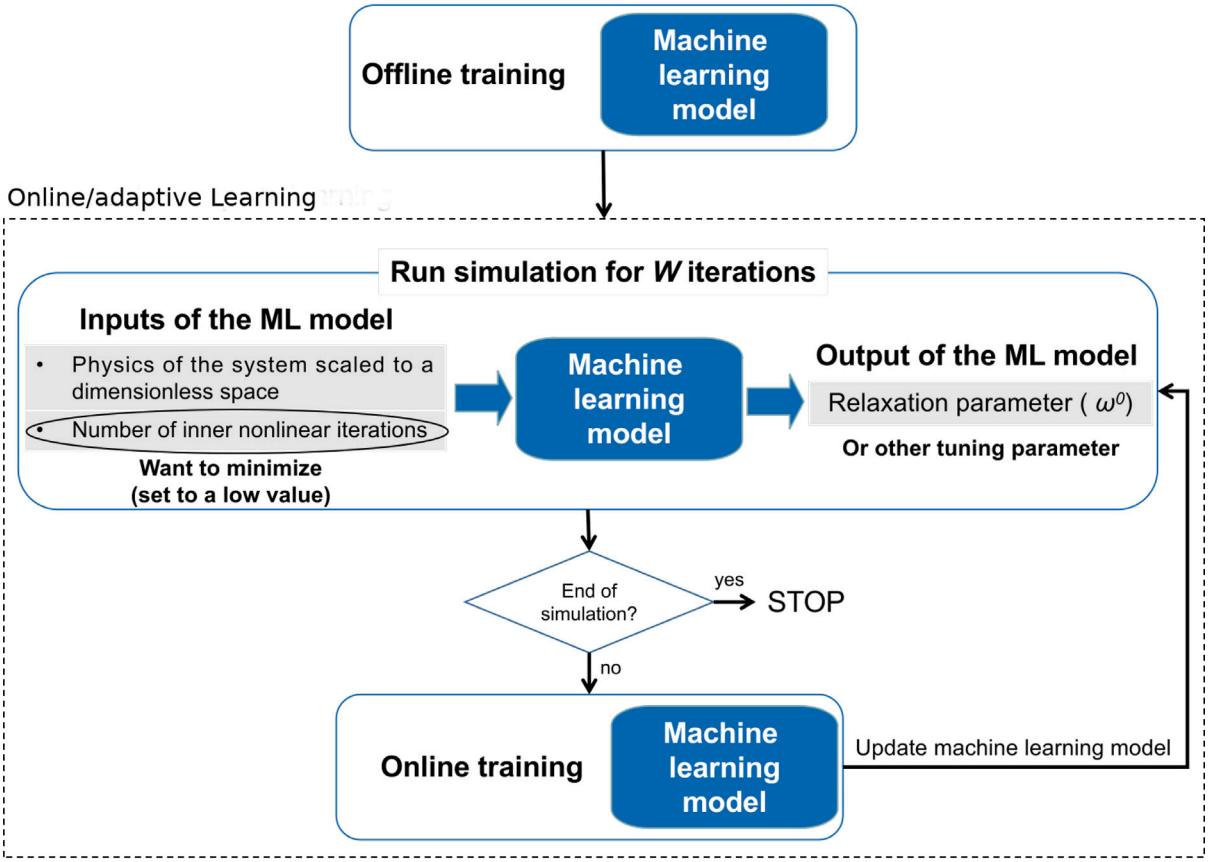


Fig. 2. Online/adaptive learning acceleration for nonlinear PDE solvers.

value of the relaxation parameter. Further details about the offline training can be found in Silva et al. (2021). Here, we test different sets of dimensionless numbers as input parameters, and we analyse several machine learning models in terms of accuracy, prediction time and simulation results.

After the offline stage, we use the machine learning model to determine the value of the numerical relaxation for a number of outer nonlinear iteration that we call W (as in Fig. 2 and Algorithm 1). Following that, we can update the machine learning model using a batch-incremental method ($W > 1$), or an instance-incremental method ($W = 1$). After updating, we run more W outer nonlinear iterations and update the machine learning model again. The process continues until the end of the numerical simulation. Given that batch-incremental and instance-incremental methods can perform similarly (Read et al., 2012), and to avoid the cost added to the numerical simulation each time we update the machine learning model, we choose to use batch-incremental methods here (or $W > 1$). We also test different batch sizes in order to determine the best one for the solver and multiphase porous media problems used here. It is worth mentioning that since we have a fixed simulation time the number of updates of the machine learning model is finite, different from classical online learning problems.

The main goal of the adaptive learning acceleration is to train the machine learning model offline using a simple reservoir model (that is fast to run). Then apply the proposed acceleration to more complex and challenge reservoirs, while still learning from them in run time. During the numerical simulation, we learn from previous nonlinear iterations of the running reservoir model in order to better calculate the relaxation factor for the coming iterations.

Algorithm 1 Online/adaptive learning acceleration pseudo-code

Require: Train the Machine Learning Model (ML) offline
$W \leftarrow$ buffer size
$c \leftarrow 0$ ▷ Counter for the outer nonlinear iterations
Start simulation:
while not end of the simulation **do**
relaxation $\leftarrow$ ML(Physics of the system, Inner nonlinear iterations set to 1)
Run one outer nonlinear iteration using *relaxation*.
$c \leftarrow c + 1$
if $c = W$ **then**
Update ML using data from the last W outer nonlinear iterations.
$c \leftarrow 0$
else
continue
end if
end while
End simulation

3.1. Input parameters and dimensionless numbers

The spatial distribution of fluids in multiphase flow through porous media are governed by the interplay of capillary, viscous, and gravitational forces (Hoteit and Firoozabadi, 2008; Li and Tchelepi, 2015; Debbabi et al., 2017a,b, 2018b). The relative significance of these mechanisms is influenced by a combination of system length-scales,

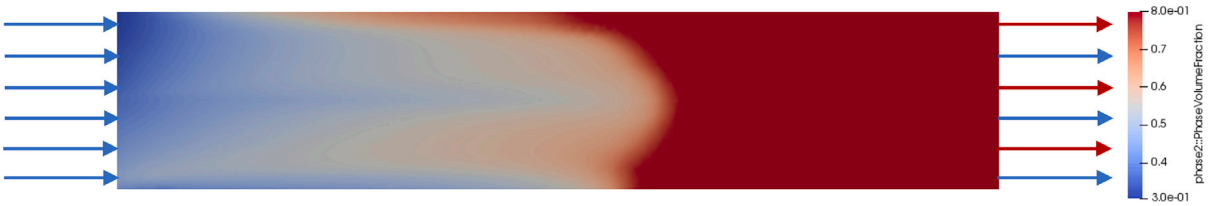


Fig. 3. Two-dimensional homogeneous layered reservoir used for offline training. It shows the saturation of the displaced phase in one point in time during the numerical simulation. Blue represents the injected phase and red the displaced phase.

Table 1
Input parameters for the machine learning model. CFL stands for Courant–Friedrichs–Lewy (CFL). The features where MIN, MAX and AVERAGE applies are defined control-volume wise.

Features	Silva et al. (2021)	Set 1	Set 2
Effective aspect ratio	One value	One value	One value
Darcy velocity	Average, max and min	Average	Average
Total mobility	Average, max and min	Average	Average
CFL number	Max value	Max value	Max value
Shock-front CFL number	Max value	Max value	Max value
Shock-front number ratio	One value	One value	One value
Shock-front mobility ratio	Average, max and min	Average	Average
Longitudinal capillary	Average, max and min	Average	Average
Transverse capillary	Average, max and min	Average	×
Buoyancy number	Average, max and min	Average	×
Longitudinal buoyancy	Average, max and min	Average	Average
Transverse buoyancy	Average, max and min	Average	×
Vanishing artificial diffusion	Average, max and min	Average	Average
Transport equation residual	One value	One value	One value
Transport equation residual old	One value	One value	One value
Transport equation residual ratio	One value	One value	One value
Inner nonlinear iterations	One value	One value	One value
Number of features:	35	17	14

fluid-rock properties, and flow rates (Debbabi et al., 2018b). Consequently, the inputs for the machine learning model are selected to be the dimensionless numbers as outlined in Debbabi et al. (2017a,b) and Debbabi et al. (2018a). Employing dimensionless parameters to calculate the relaxation parameter enables the use of a simple two-dimensional layered reservoir for training, while effectively covering a broad spectrum of the physical space. All the input parameters are summarized in Table 1. A detailed description of the feature selection is given in Section 5.1.

3.2. Training dataset

We run the offline training using data from a multiphase porous media flow simulation of a simple two-dimensional layered model, as shown in Fig. 3. We inject one phase on the left side and produce both phases on the right side. The aim is for the injected phase to push the displaced phase through the porous media. Because the reservoir model must be run numerous times to generate the dataset for training, employing a simple reservoir model considerably simplifies the training process.

To create a diverse offline training dataset, we perturb the physical parameters listed in Table 1 by varying the simulation inputs. These inputs include horizontal and vertical permeabilities, porosity, time step size, capillary pressure, relative permeability, gravity magnitude and direction, viscosity, and the relaxation parameter. We run 6500 simulations, which represents approximately 1.8 million instances (outer nonlinear iterations). This dataset is split into a training (80%) and test (20%) set.

In the online stage (online/adaptive learning), for each outer nonlinear iteration we store: the dimensionless numbers in Table 1, the relaxation factor, and the number of inner nonlinear iterations. After running W outer nonlinear iterations, we then run the online training. We repeat this process every W nonlinear iterations until the end of the numerical simulation.

Table 2
Rock-fluid modelling using the Brooks–Corey parametrization.

Property	Cases 1 and 2	Case 3	Case 4
Relative permeability end point of water	1.0	0.3	0.3
Relative permeability end point of oil	1.0	0.8	0.8
Relative permeability exponent	2.0	2.0	2.0
Immobile fraction of water	20%	20%	20%
Immobile fraction of oil	30%	20%	20%
Entry capillary pressure	1000 Pa	100 Pa	10000 Pa
Capillary exponent	1.0	1.0	1.0

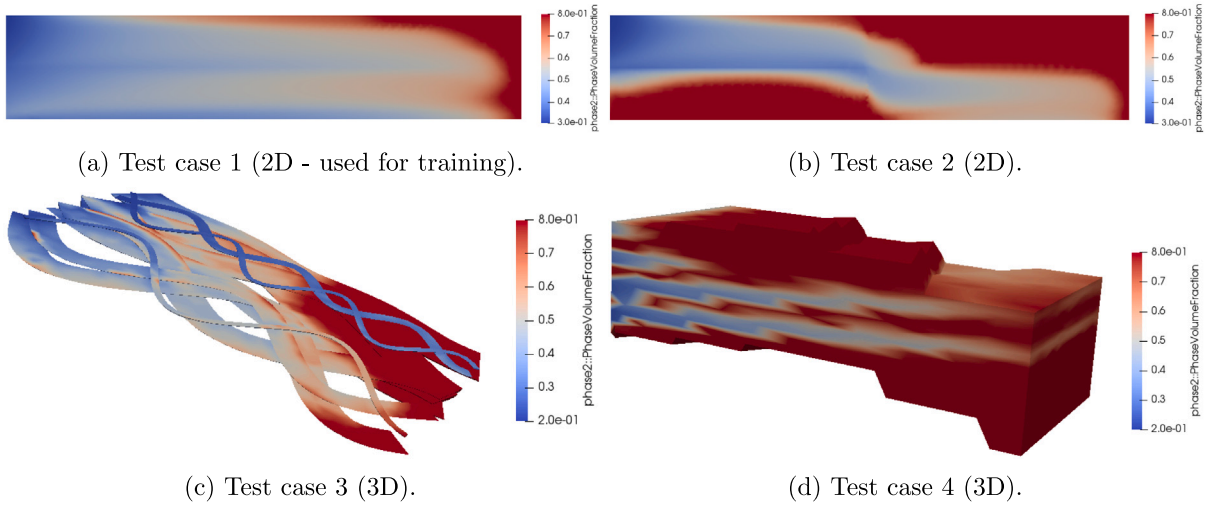
4. Test case description

We evaluated the adaptive learning acceleration across four distinct reservoir models as in Silva et al. (2021). Each of them represents two-phase immiscible flow within a porous medium:

1. Test case 1: two-dimensional layered reservoir model used for training.
2. Test case 2: heterogeneous two-dimensional reservoir model with distinct permeabilities across its four quadrants.
3. Test case 3: more intricate three-dimensional reservoir model simulating channelized formations.
4. Test case 4: complex three-dimensional reservoir model with faulted structures.

For all reservoir models, both gravity and capillary pressure were considered. The capillary pressure and relative permeability were modelled using the Brooks–Corey parametrization (Brooks and Corey, 1964) with the values shown in Table 2. In our tests, we injected one phase from the left and extracted both phases from the right. Fig. 4 provides a saturation map for each test case.

The convergence criteria for the nonlinear solver are defined as follows: the relative mass conservation of the system must be less



(a) Test case 1 (2D - used for training). (b) Test case 2 (2D).

(c) Test case 3 (3D). (d) Test case 4 (3D).

Fig. 4. Saturation of the displaced phase in one point in time during the numerical simulation. Blue represents the injected phase and red the displaced phase. In all cases, we injected one phase on the left and produce both phases on the right.

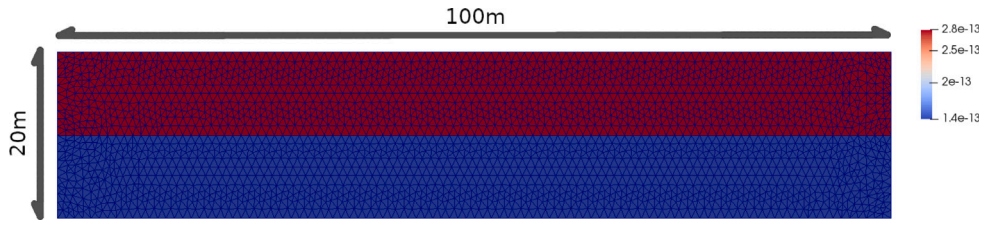


Fig. 5. Test case 1 (2D) permeability field in m^2.

than 10^{-3}, and the infinite norm of the delta saturation between two successive nonlinear iterations should be less than 10^{-2} within a single time step. Additionally, we set maximum limits of 30 and 10 for the outer and inner nonlinear iterations, respectively. In evaluating different scenarios, the total number of nonlinear iterations is determined by summing the outer iterations and one-third of the inner iterations. This approach aligns with the methodologies previously established by Salinas et al. (2017a) and Silva et al. (2021).

4.1. Test case 1

Test case 1 is the layered reservoir model used to generate the dataset for the offline training. The model comprises two layers and is built in two-dimensions (Fig. 5). The upper layer displays a porosity of 10% and a horizontal permeability measuring 200 millidarcies (mD). Conversely, the lower layer possesses a porosity of 20% and a horizontal permeability of 100 mD. The vertical permeability in both layers is 10% of their respective horizontal permeabilities. Additionally, the viscosity ratio of the injected fluid to the displaced fluid is 5 and the density contrast between these two fluids is 300 kg/m^3.

4.2. Test case 2

Test case 2 is a heterogeneous two-dimensional reservoir model with distinct permeabilities across its four quadrants (Fig. 6). In the upper layer, the horizontal permeabilities are 200 mD and 20 mD, from left to right. In contrast, the lower layer exhibits horizontal permeabilities of 10 mD and 100 mD. Across both layers, the vertical permeability equals the horizontal permeability. Additionally, a uniform porosity of 20% is observed across all layers. Regarding fluid properties, a viscosity ratio of 5 exists between the displaced and the injected fluid, and there is a density contrast of 300 kg/m^3 between fluids.

4.3. Test case 3

Test case 3 is a three-dimensional reservoir model featuring fluvial sandstone channels situated within a backdrop of low-permeability mudstone (Fig. 7). The channels are classified into three types based on their width: thin, medium, and wide. The permeabilities assigned to these types are 1000 mD for thin channels, 200 mD for medium-sized channels, and 100 mD for wide channels. In each channel type, the permeability in the vertical direction matches the horizontal permeability. The porosity for all channels is uniformly set at 20%. Further parameters include a viscosity ratio of 5 between the displaced and injected fluid, and a density contrast of 289 kg/m^3 between fluids.

4.4. Test case 4

Test case 4 features a three-dimensional, complex reservoir model characterized by faulted geological structures (Fig. 8). The reservoir is characterized by a sequence of alternating layers of mudstone and sandstone. The mudstone layers have a horizontal permeability of 1 mD, while the sandstone layers have a horizontal permeability of 1000 mD. In both layer types, the permeability in the vertical direction matches the horizontal permeability. Regarding porosity, the mudstone layers exhibit a porosity of 20%, while the sandstone layers display a lower porosity of 10%. Moreover, a viscosity ratio of 5 is present between the displaced and the injected fluid accompanied by a phase density contrast of 300 kg/m^3.

5. Numerical results

In the following sections, we test and select the best set of dimensionless numbers as input parameters, and analyse several machine learning models in terms of accuracy, prediction time and simulation results. Following that, we apply and analyse the proposed

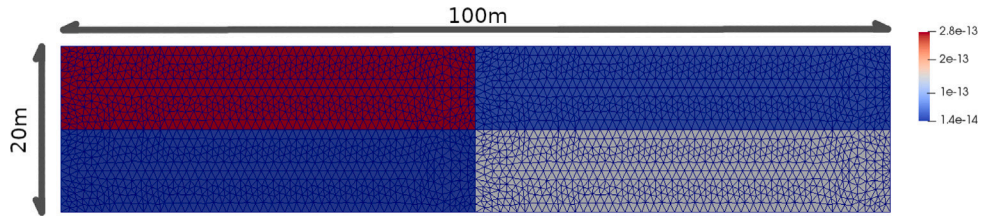


Fig. 6. Test case 2 (2D) permeability field in m^2.

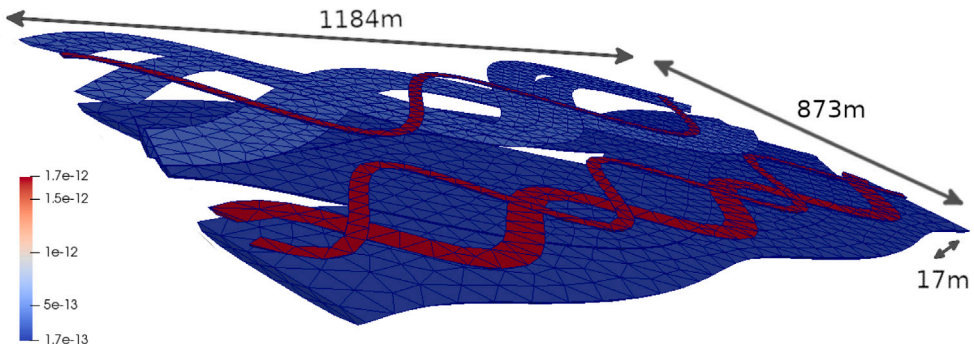


Fig. 7. Test case 3 (3D) permeability field in m^2.

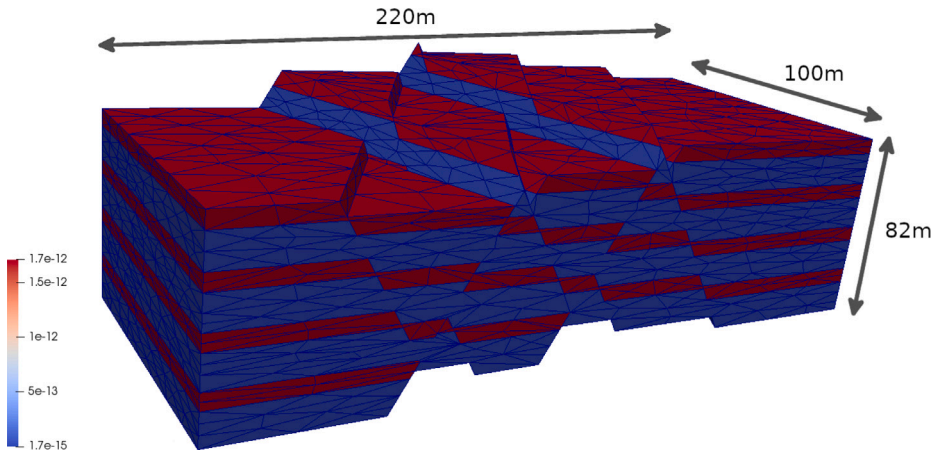


Fig. 8. Test case 4 (3D) permeability field in m^2.

online/adaptive learning acceleration. Finally, we integrate the best machine learning model into IC-FERST in order to actually generate a walltime reduction. As a baseline for comparison, we use here the nonlinear solver acceleration presented in our previous work (Silva et al., 2021). There we used a random forest to calculate the relaxation parameter, but relying only on the offline training. For the remainder of this paper, the random forest model used in Silva et al. (2021) will be referred as original random forest.

5.1. Machine learning feature selection

We investigate different sets of features, in order to evaluate their effect on the proposed method. In Silva et al. (2021) a set of 35 features were chosen (see Table 1), and for the parameters defined in each grid cell the average, minimum and maximum over the domain were calculated, excepted for the Courant–Friedrichs–Lewy (CFL) number where it is already known that the maximum value plays an important role in the nonlinear solver convergence (Younis et al., 2010; Salinas et al., 2017a; Obeysekara et al., 2021). In this work, we test two new sets of features as shown in Table 1. Fig. 9 shows the correlation between all features, calculated from the numerical simulations in the training dataset. We notice from the figure that the average, minimum and maximum values are highly correlated when considering the same parameter. Therefore, in the first set (Set 1) we consider only the average. For the second set (Set 2), we also see that capillary and buoyancy numbers are highly correlated among themselves. Hence, we have looked the feature importance (see Fig. 10), and for the capillary and buoyancy numbers we included only the most important ones. As for the original random forest, we run a hyperparameter optimization using a grid search with a 3-fold cross-validation for the two new sets of features. The implementation of the random forests used here is the one in Pedregosa et al. (2011). For the sake of comparison, the results reported in this section only consider the offline training.

In Fig. 10, we show the feature importance of the SET 1 generated by the original random forest. We can notice that the transport equation residual ratio (how fast the solver is converging) is the most important feature, while the other residuals also appear among the top variables. Moreover, we see that the Shock-front CFL number and CFL number appear among the most important features. This CFL dependence on the solver convergence is also seen in Younis et al. (2010), Salinas et al. (2017a) and Obeysekara et al. (2021). Fig. 10 also shows that the most important features among the buoyancy and capillary numbers are the

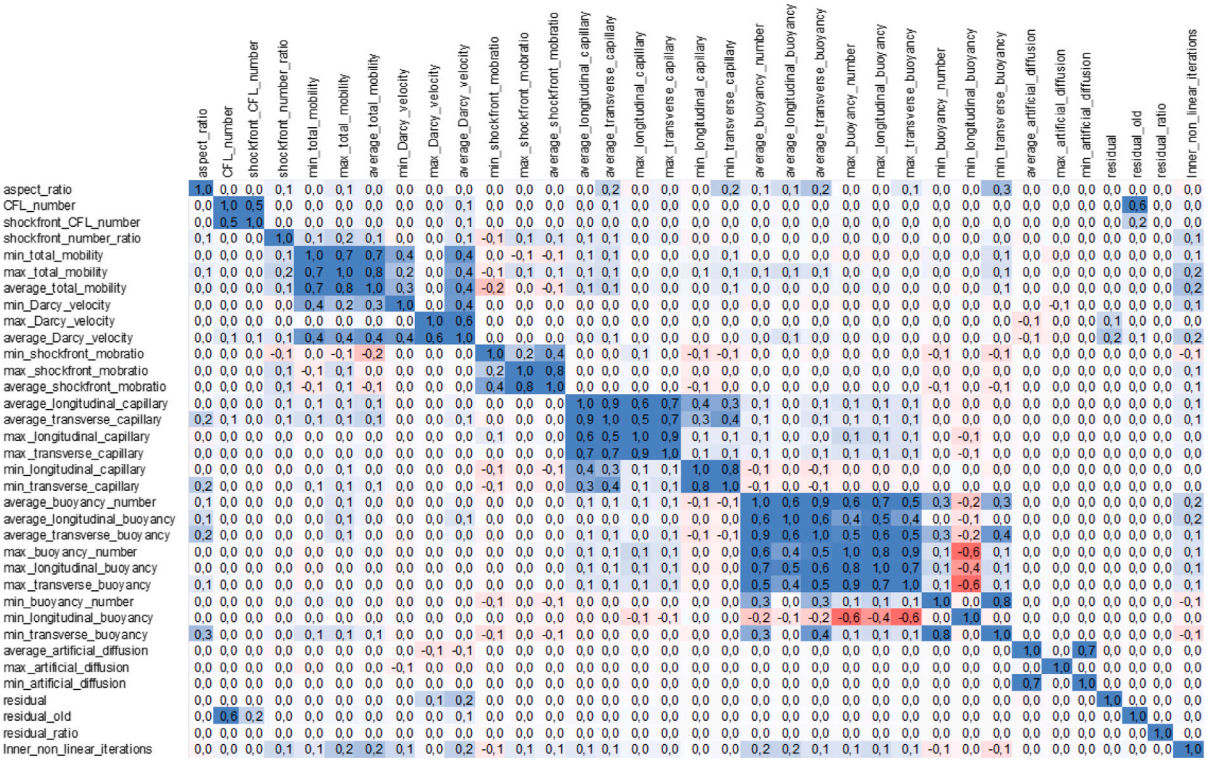


Fig. 9. Correlation between features calculated using the numerical simulations in the training dataset.

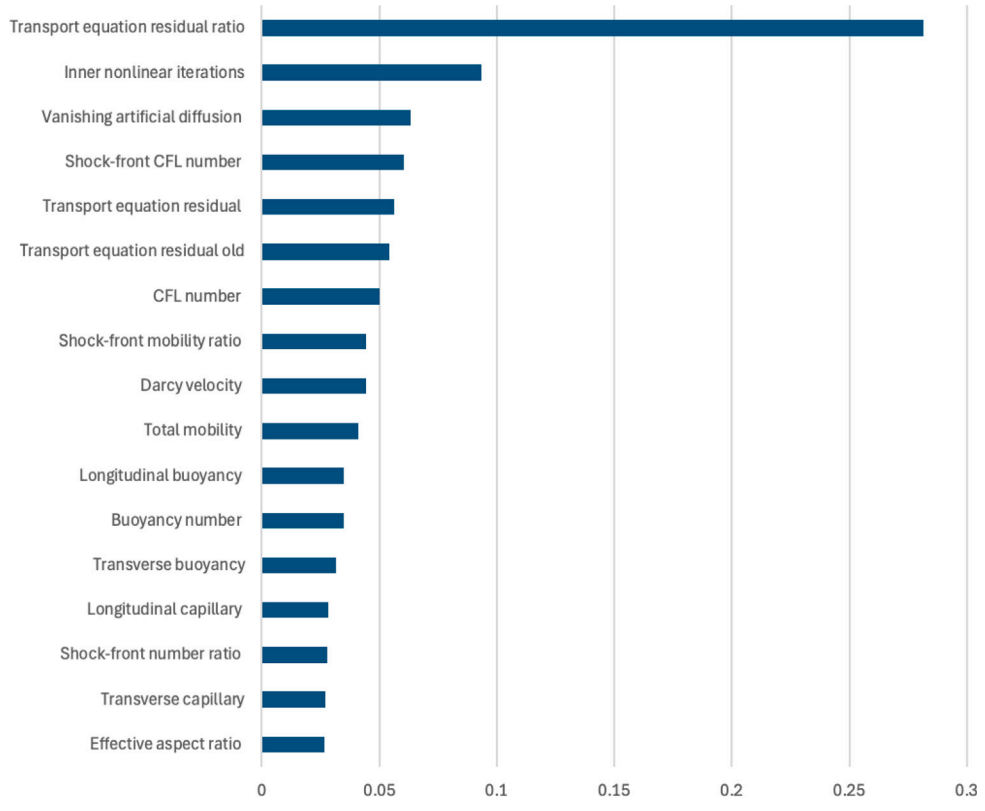


Fig. 10. Feature importance of SET 1.

longitudinal buoyancy and longitudinal capillary. Because of that, we kept them on SET 2, as mentioned in the previous paragraph.

Fig. 11 shows a comparison of the different number of features. In Fig. 11(a) we can see the test root mean square error (RMSE) of the random forest models for the three sets of features. There is no significant difference in the RMSE when comparing the models. However, in Figs. 11(b) and 11(c) we can notice that the reduction in the number of nonlinear iterations (improvement) is greater for Set 1

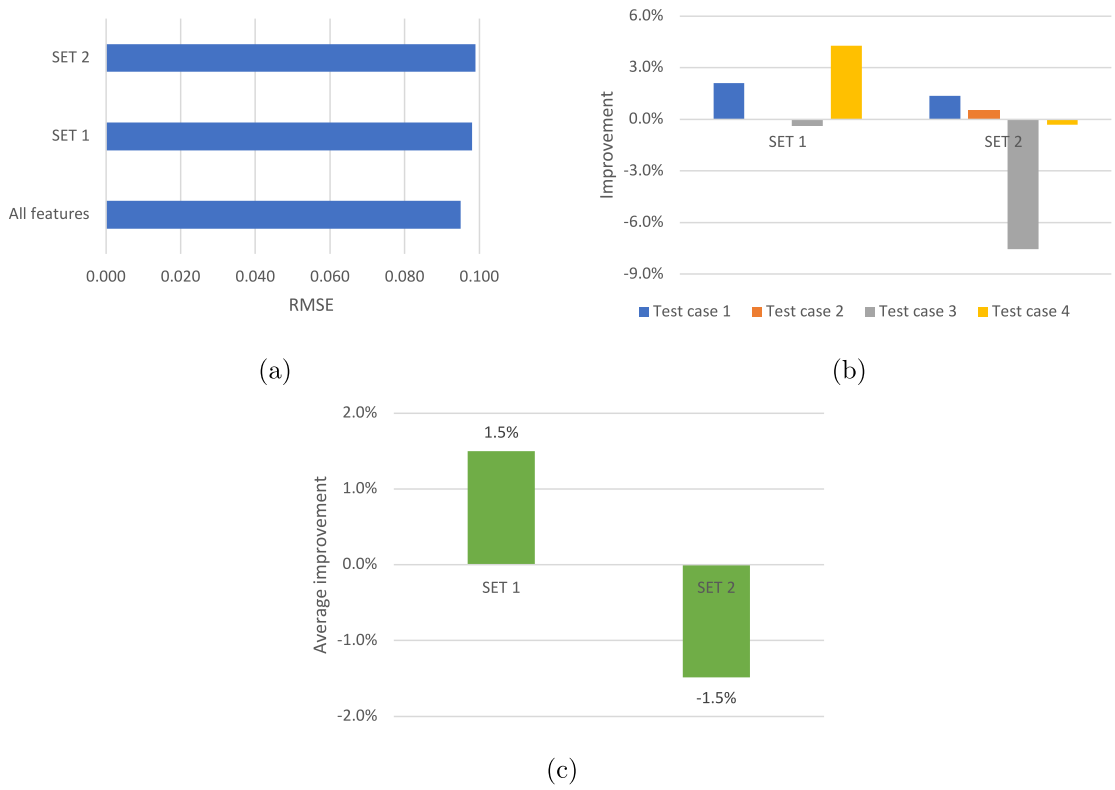


Fig. 11. Comparison of different number of features. (a) the test root mean square error of the trained random forest models. (b) the improvement in the number on nonlinear iterations compared with the original random forest. (c) average improvement (reduction) in the number of nonlinear iterations compared with the original random forest.

(fewer features). Set 2 (many fewer features) increased the average number of nonlinear iterations. The results show that the acceleration using the random forest with Set 1 performed better than the original random forest and the one with Set 2. Therefore, the 17 features of Set 1 were selected to continue the experiments.

5.2. Machine learning model selection

We compare several machine learning models for controlling the numerical relaxation. All of them were implemented using Pedregosa et al. (2011), except for the xgboost models where we used Chen and Guestrin (2016) and the neural network (multilayer perceptron - MLP) where we use Abadi et al. (2015). The input in all the models are the 17 features selected in the previous section (Set 1), except for the original random forest that uses the 38 features in Table 1. For each machine learning model we run a hyperparameter optimization using a grid search with a 3-fold cross-validation. Again, for the sake of comparison, the results reported in this section only consider the offline training. In the following, we give a brief description of each machine learning method tested.

- **Linear regression**: Linear regression estimates the output by computing a linear combination of the input features along with an intercept term (Hastie et al., 2009; Géron, 2019). This results in a model that represents a linear relationship between the inputs and the target variables.
- **Decision tree**: Decision trees make predictions using a hierarchical structure of nodes and branches, where the data is split into distinct rectangular regions based on feature values. Each region is assigned a fixed output value. Although simple in design, decision trees are highly effective and form the basis for more sophisticated ensemble methods such as random forests and gradient boosting (Hastie et al., 2009; Géron, 2019).
- **Random forests**: Random forests, first proposed by Breiman (2001), are ensembles of decision trees. During training, each tree is built using a different bootstrap sample of the data, and at each split, a random subset of features is considered to determine the best split. These random sources help reduce the variance of the model (Breiman et al., 1998; Breiman, 2001). The final output is obtained by averaging the predictions from all individual trees.
- **Extra tree**: Extremely randomized trees are an ensemble learning method based on decision trees that introduces additional randomness in how splits are determined, compared to Random Forests. This added randomness typically helps to lower the model variance, although it may slightly increase bias (Geurts et al., 2006).
- **Adaboost**: Adaptive Boosting is an ensemble learning technique that combines multiple weak learners into a single strong classifier. Unlike bagging methods, AdaBoost builds models sequentially, where each new model focuses more on the instances that previous models misclassified. This iterative reweighting of training samples helps reduce bias and improve overall accuracy (Freund and Schapire, 1997).
- **Xgboost**: eXtreme Gradient Boosting is an advanced implementation of gradient-boosted decision trees (GBDT) (Chen and Guestrin, 2016). GBDT constructs decision trees in a sequential manner, where each new tree aims to correct the errors made by the previous ones. Xgboost enhances this approach by incorporating techniques such as regularization, which contribute to its strong performance across a wide range of datasets (Shwartz-Ziv and Armon, 2022; Borisov et al., 2022; Grinsztajn et al., 2022). It has become one of the most popular models for tabular data. The xgboost library (Chen and Guestrin, 2016) also offers a variant of random forests. In this study, we evaluate both the standard xgboost model and its random forest counterpart.

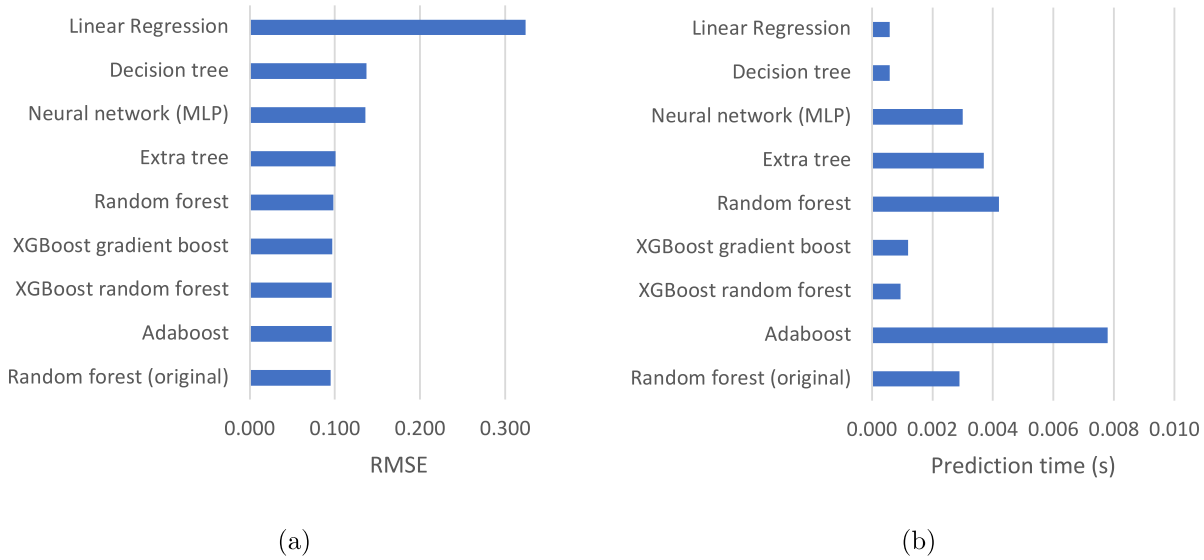


Fig. 12. Comparison of different machine learning models. (a) the test root mean square error. (b) the one-instance prediction time.

- **Multilayer perceptron**: A Multilayer Perceptron (MLP) is a fully connected, feed-forward neural network consisting of multiple layers of neurons, each employing nonlinear activation functions. The architecture comprises at least three distinct layers: an input layer, one or more hidden layers, and an output layer (Goodfellow et al., 2016; Hastie et al., 2009).

Fig. 12 shows a comparison of the different models. In Fig. 12(a) we can see the test RMSE of the machine learning models. Apart from the linear regression all the models present similar RMSE. Fig. 12(b) shows the prediction time for one instance. For each machine learning model, we run the prediction 1000 times and average the resulted prediction time. As we use the machine learning model to generate the relaxation parameter in each outer nonlinear iteration, we want the prediction time to be as low as possible. We can notice that the xgboost models, linear regression and decision tree exhibit the lowest prediction times.

Fig. 13 shows the reduction in the number of nonlinear iterations (improvement) for all the machine learning models, using as baseline the original random forest. We can also see the cumulative number of nonlinear iterations during the numerical simulation for all the machine learning models and the case with no relaxation in Fig. 14. In Fig. 13(a) the solutions accelerated using the adaboost and decision tree have failed to converge for test case 4. Because of that, their average improvement is not shown in Fig. 13(b). The only models that have generated better results than the original random forest are the xgboost random forest and the random forest (both with Set 1). It is also worth noting (from Fig. 14) that the approach with no relaxation generated the worst results for the test cases 1 and 2, and it failed to converge for the test cases 3 and 4.

5.3. Online learning

In the previous sections, we focused on the offline training. The two machine learning models that generated better results than the original random forest are the xgboost random forest and the random forest. Adding that ensemble learners have the advantage of being flexible, as new learners can be selectively added or removed, that they are usually easier to optimize, and that they are often used when learning from evolving data streams (Oza and Russell, 2001; Read et al., 2012; Gomes et al., 2017a; Montiel et al., 2020), we have selected these two models to test the online/adaptive learning shown in Fig. 2.

Many state-of-the-art ensemble methods for data stream learning are adapted versions of bagging and boosting strategies (Gomes et al., 2017a; Montiel et al., 2020). In this work, given that the xgboost library (Chen and Guestrin, 2016) already supports a boosting strategy for training continuation, we apply the online learning method Adaptive XGBoost, proposed by Montiel et al. (2020), to the xgboost model. Nonetheless, we consider here that we already have the first learner trained offline. In the online stage, as new outer nonlinear iterations arrive, they are stored in a buffer of size W. Once the buffer is full, we train a new member of the ensemble with the residuals from the previous members. It is worth noting that we do not reach the maximum ensemble size, unlike Montiel et al. (2020), because the time of the numerical simulation is finite the same way as the number of updates (usually less than 10 updates by numerical simulation). For the random forest, we modify the method proposed by Montiel et al. (2020) to work as a bagging strategy, since this is the strategy supported by the random forest in the Scikit-learn library (Pedregosa et al., 2011). As in the Adaptive XGBoost method, as new outer nonlinear iterations arrives, they are stored in a buffer of size W. The difference in the random forest is that once the buffer is full, we train a new set of ensemble members with the samples in the buffer, and add it to the previous ensemble. The hyperparameters used to configure this methods are described in Appendix.

We applied these two strategies to the four test cases shown in Fig. 4. Tables 3 and 4 show the percentage reduction (improvement) in the number of nonlinear iterations compared with the original random forest (state-of-the-art results from Silva et al. (2021)). The columns RF and XGBRF represent the two machine learning models selected from the previous section, but without the online learning stage. In the batch-incremental learning, the size of the batch must be chosen to provide a balance between accuracy (large batches) and response to new instances (smaller batches) (Read et al., 2012). In this context, we test different buffer sizes (W) represented in the remaining columns of Tables 3 and 4. Fig. 15 also shows the average reduction of each one of these strategies. We also notice that the strategy with buffer size equals 50 ($W = 50$) produced the best average performance for both machine learning models. Also for the strategy with buffer size equals 25 ($W = 25$), the number of nonlinear iterations became worse than the original random forest for some test cases. This can indicate that updating the machine learning model with short buffer sizes does not necessarily improve the final results.

Fig. 16 shows the comparison of the number of inner nonlinear iterations over the simulation period, for the test case 1. The horizontal axes represent the outer nonlinear iterations. We can notice from the blue box on the left, that for the first 50 outer nonlinear iterations the

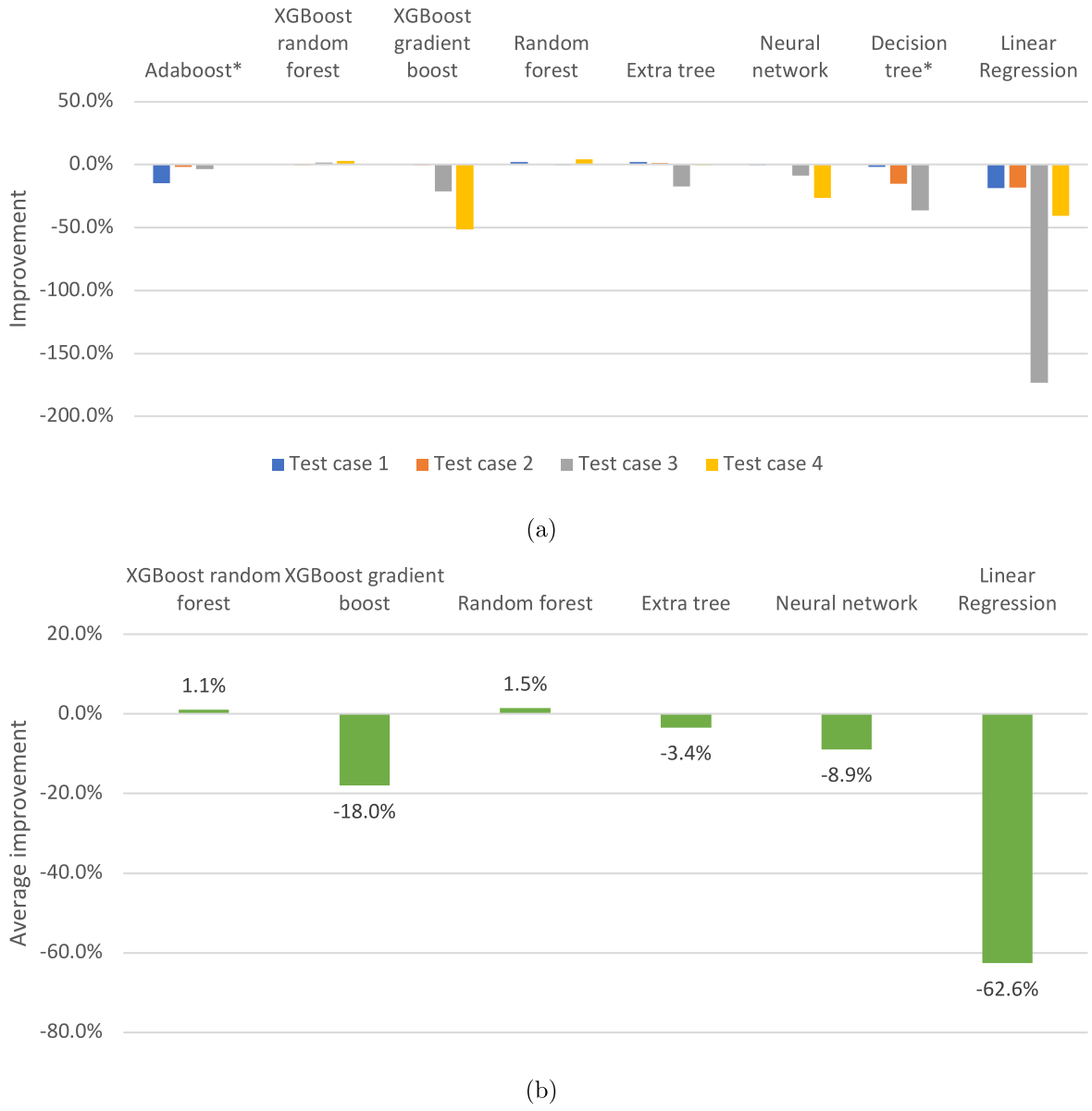


Fig. 13. Comparison of different machine learning models. (a) Improvement in the number on nonlinear iterations compared with the original random forest. (b) Average improvement (reduction) in the number of nonlinear iterations compared with the original random forest.

Table 3
Adaptive learning acceleration using the random forest, and a bagging strategy for the online learning. Improvement (reduction) in the number on nonlinear iterations compared with the original random forest.

Test case	RF	$W = 25$	$W = 50$	$W = 200$
Case 1	2.1%	2.8%	3.7%	1.2%
Case 2	0.0%	−1.8%	0.6%	0.5%
Case 3	−0.4%	2.7%	0.9%	2.7%
Case 4	4.3%	5.0%	4.5%	4.2%
Avg. reduction	1.5%	2.2%	2.4%	2.1%

Table 4
Adaptive learning acceleration using the xgboost random forest, and a boosting strategy for the online learning. Improvement (reduction) in the number on nonlinear iterations compared with the original random forest.

Test case	XGBRF	$W = 25$	$W = 50$	$W = 200$
Case 1	0.5%	−0.4%	2.4%	0.8%
Case 2	−0.6%	−1.3%	0.0%	0.0%
Case 3	1.8%	−1.3%	0.4%	0.4%
Case 4	2.8%	3.4%	3.3%	2.9%
Avg. reduction	1.1%	0.1%	1.5%	1.0%

results from the random forest with the online/adaptive learning and the one without it are the same. This is because the buffer has size 50 ($W = 50$), then until it is full no update is performed in the machine learning model. We can also see a clear reduction in the inner nonlinear iterations from the original random forest to the online/adaptive learning. It is worth noting that approximately after 300 outer nonlinear iterations the shock front (the interface between the two phases in the porous media, see Fig. 3) has passed the simulation domain, making the nonlinear solver convergence much easier for the three strategies, only one inner nonlinear iteration is required to reach the convergence criteria. In Fig. 17, we show the values of the relaxation parameter over the outer nonlinear iterations for the random forest with and without online/adaptive learning. It shows that the case with the online/adaptive learning reduces the high-frequency variations in the relaxation. During the online learning, the sample training space comprises only

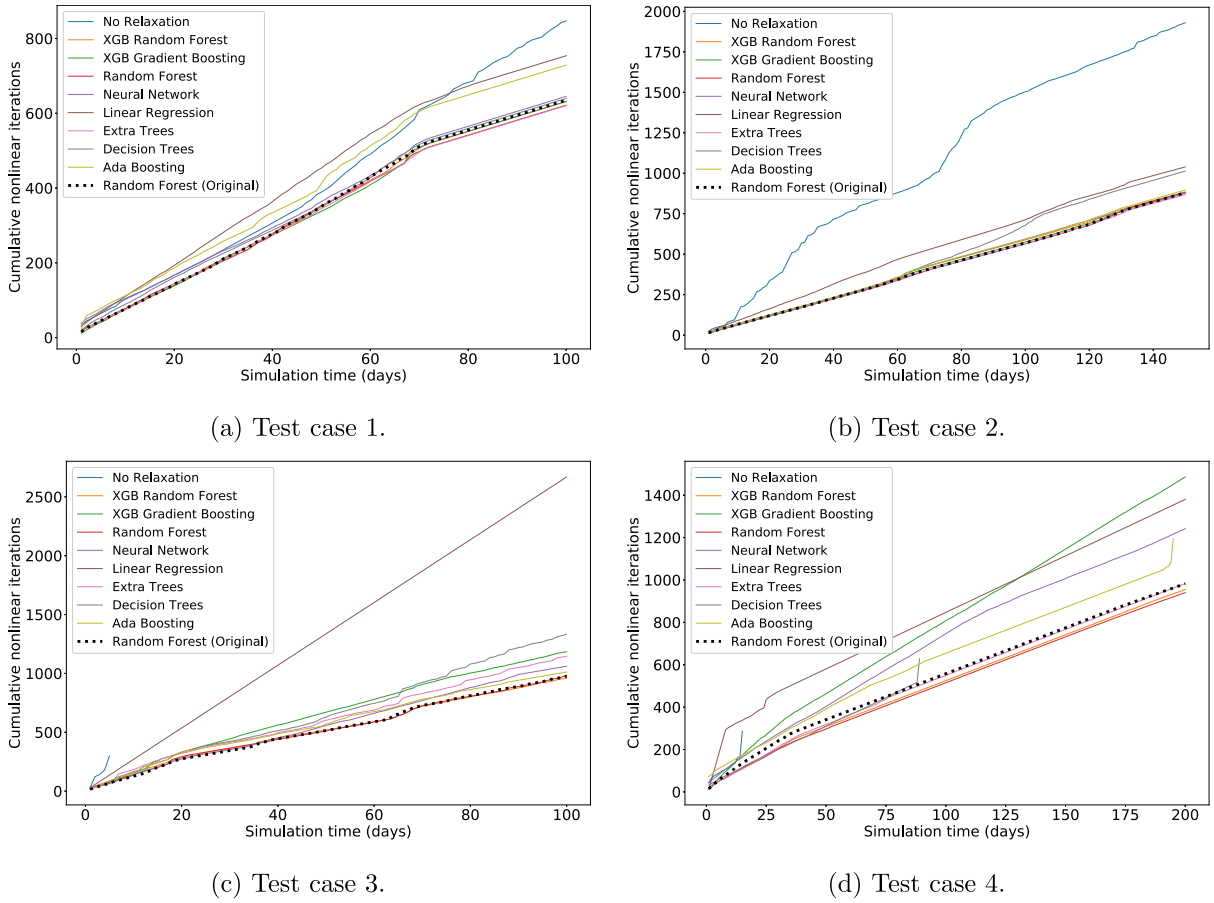


Fig. 14. Cumulative number of nonlinear iterations for different machine learning models and for the case with no relaxation.

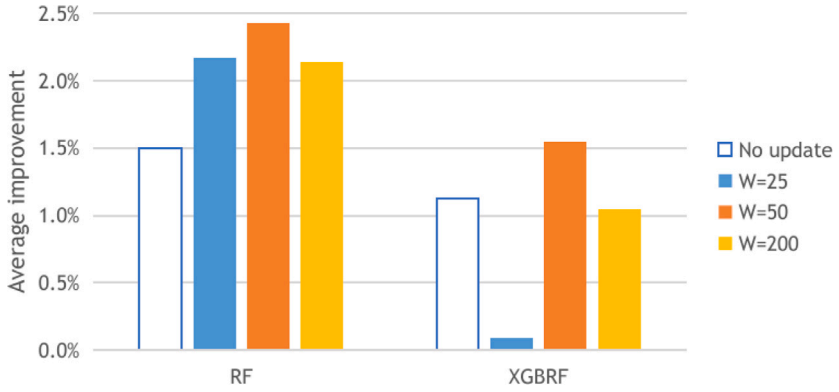


Fig. 15. Average improvement (reduction) in the number of nonlinear iterations compared with the original random forest. The Base case is considering no online updating.

the previous iterations, this maybe guiding the machine learning model to choose the best relaxation within this range.

5.4. Direct coupling

All of the previous machine learning models, including the original random forest (Silva et al., 2021), were not properly coupled with the numerical simulator in Fortran. An external Python script was called in each outer nonlinear iteration to generate the relaxation factor and it was passed "on the fly" to the numerical simulator. For that reason, no actual reduction in walltime was observed. Analysing the prediction time (Fig. 12(b)), the reduction in the number of nonlinear iterations (Fig. 13), and considering the viability to integrate the machine learning code with the numerical simulator (IC-FERST) in Fortran, we chose the xgboost random forest to perform the coupling. A Fortran API for the xgboost was developed (available at https://github.com/viluiz/solver/tree/main/xgboost_coupling), and we were able to load the machine learning model in memory and call it in each outer nonlinear iteration using a Fortran code. Although, the online training of the machine learning model was not yet supported in the API, thus the results in this section consider $W = \infty$, which means no online update (the online update would further improve the results). Table 5 shows the reduction in the number of nonlinear iterations and walltime, when we compare the proposed method with the default dynamic relaxation presented in IC-FERST (Salinas et al., 2017a). It is a dynamic relaxation that adjusts the relaxation parameter mainly based on the CFL number. It has demonstrated great improvements in the convergence rate of the solver for a wide range of cases (Salinas et al., 2017a), and up to this

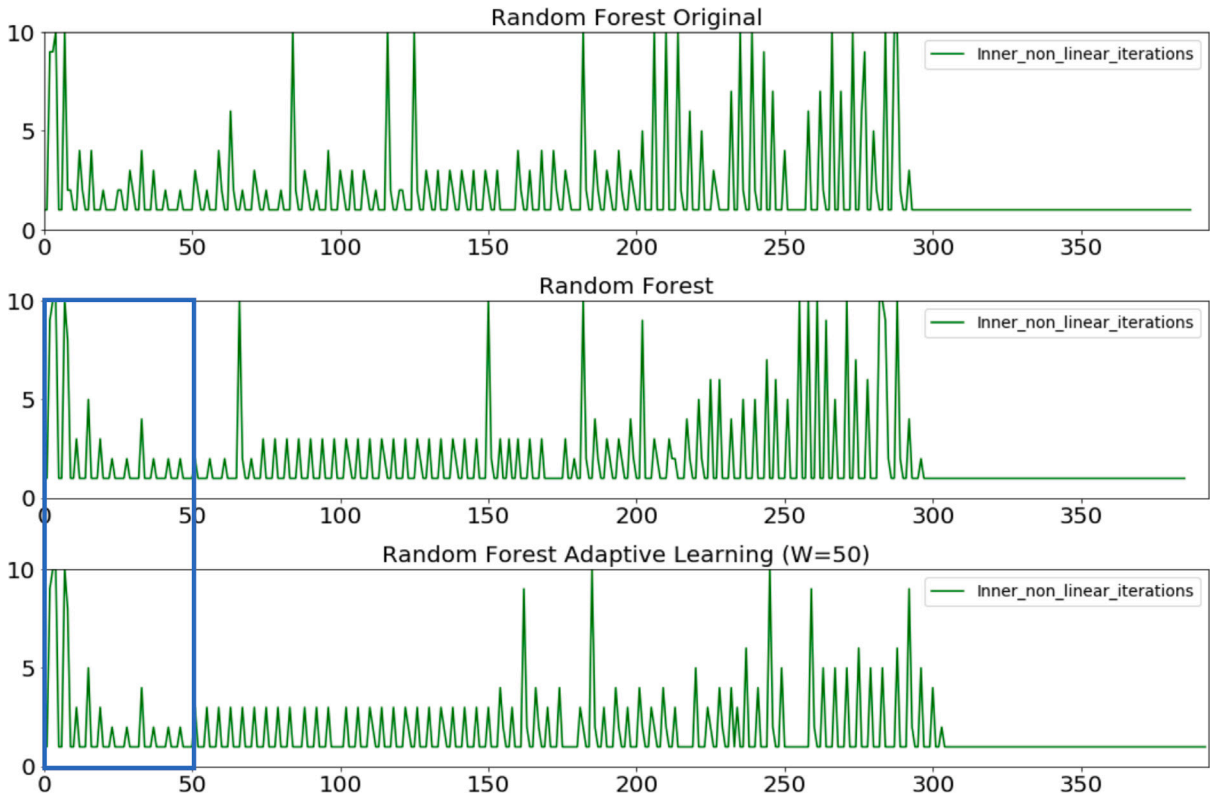


Fig. 16. Comparison of the number of inner nonlinear iteration over the simulation period, for test case 1. The horizontal axes are the outer nonlinear iterations and the vertical axes the number of inner nonlinear iterations.

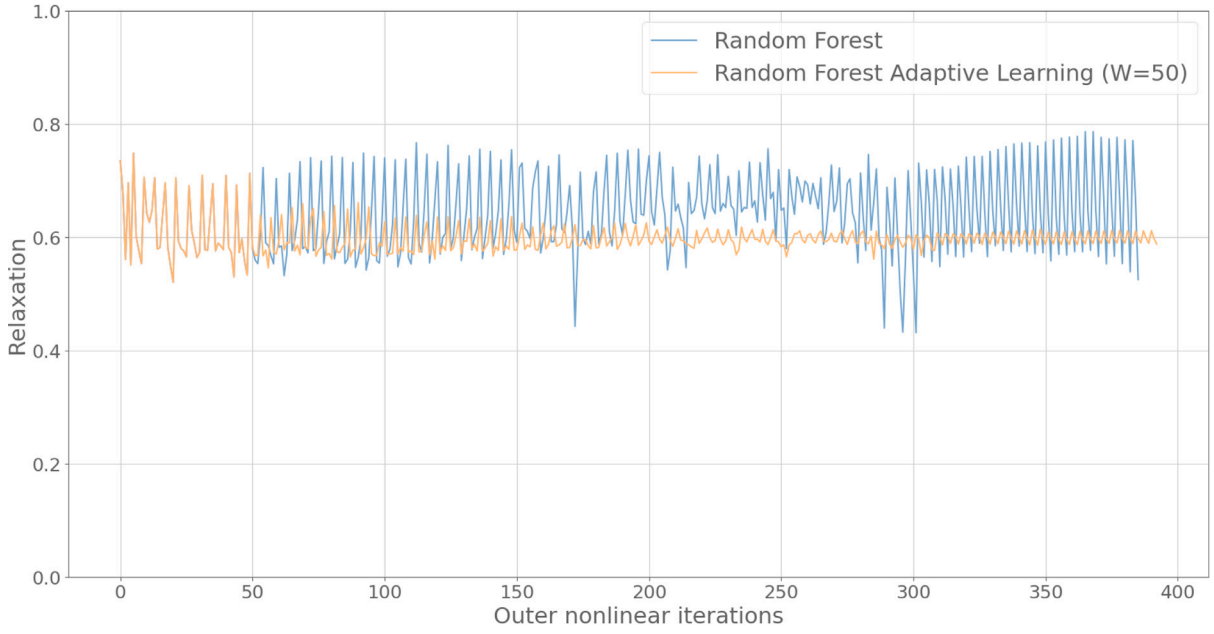


Fig. 17. Values of the relaxation parameter over the simulation period, for test case 1.

point was the best available approach to calculate the relaxation in ICFERST. The method proposed here is able to outperform this approach, showing actual run time improvements. The results in Table 5 show that the reduction in walltime is, in some cases, even greater than the reduction in the number of nonlinear iterations. It means that for some cases, the proposed method not only reduces the number of nonlinear iterations, but also makes faster the solution of each nonlinear iteration.

6. Discussion

Dealing with nonlinearities in systems of partial differential equations is always challenging and time-consuming, with a high impact on the overall applicability and performance of the nonlinear solver. Developing an efficient and robust nonlinear solver is nontrivial and requires a good understanding of the underlying physics of the system

Table 5
Reduction in run time and number of nonlinear iterations. Comparison between the proposed method (with no online update) and the default dynamic relaxation presented in IC-FERST (Salinas et al., 2017a).

Case	Run time	Nonlinear iterations
Test case 1	19%	13%
Test case 2	34%	37%
Test case 3	9%	10%
Test case 4	85%	81%

being modelled. The presented technique shows a promising route towards creating a robust and efficient nonlinear solver easily by using a machine learning approach in combination with a numerical relaxation. The approach is simple to develop and quick to implement as it requires only three steps:

- Identify dimensionless numbers and the parameter space to study.
- Train a machine learning model offline using a simple numerical model to generate the training set.
- Apply the online learning to update the machine learning model during run time.

We applied the proposed method to two and three-dimensional multiphase porous media flow problems, as these types of problems represent significant applications in science and engineering, and are very challenging to model due to complex nonlinearities. A great deal of effort has been put in over the years by academia and industry to address these problems (Li and Tchelepi, 2015; Salinas et al., 2017a; Jiang and Tchelepi, 2019; Freitas et al., 2020; Silva et al., 2021; Kadeethum et al., 2022), for which simulation time and computational resources are usually the main constraints. We tested the method in two and three-dimensional cases with very different geometries, such as layers, channel and faults, and also with a considerable range of permeability values. The proposed method can also be applied to other engineering and physical systems as long as they use a numerical relaxation (or other tuning parameter) to control the solver convergence.

The main limitations of the presented technique is the time required to train the machine learning model online and the necessity to calculate the dimensionless numbers in each nonlinear iteration. However, usually the cost of the numerical simulation is by far more expensive than the training operations required, and the mathematical calculations used to compute the dimensionless parameters. Furthermore, we used batch-incremental learning techniques instead of instance-incremental methods in order to minimize the online training cost, and the chosen set of dimensionless numbers reduces the number of calculated features when compared to Silva et al. (2021). In terms of machine learning model, we showed that the random forest outperformed the other methods for both offline and online learning. Moreover, for the online learning, a consistent improvement is achieved when using the random forest with a bagging strategy. The xgboost random forest with an online boosting strategy did not show the same consistency (Fig. 15).

It is worth mentioning that the improvement brought by the online learning (Tables 3 and 4) is smaller than the offline training (Table 5), although in terms of computational cost it is still relevant. We can see from Table 5, a direct relationship between the number of nonlinear iterations and the run time, which means that almost any reduction in the solver iterations can shortening computational time. The smaller improvement of the online learning maybe related to the effectiveness of the offline training to learn the optimal relaxation, given the online learning a small room for improvement. In this line, we can also notice that the greatest reductions in the number of nonlinear iterations and walltime (Tables 3, 4, and 5) are obtained in test case 4. It is the realistic three-dimensional faulted reservoir model and is the most challenging case from a numerical standpoint, since it has the highest capillary value and comprises several intercalated layers of low and high permeability. Capillary pressure is known for introducing a high non-linearity in the system of equations, as it introduces an effect on pressure due to the saturation (Salinas et al., 2020). This can potentially have a big impact in the non-linear solver, leading to smaller time-steps or even inability to solve the system. Likewise, high contrasts in permeability may affect the stability of the non-linear solver as the velocity of the fluid may change orders of magnitude when changing from a low permeable region to a high permeable region, and vice-versa. Thus, the room for improvement in these cases is bigger than in other scenarios where non-linear solvers do not struggle as much. The presented approach is able to mitigate the damaging effects introduced by capillary pressure and high permeability contrasts and keep the non-linear solver working efficiently.

7. Conclusion

We propose an online learning acceleration for nonlinear PDE solvers. The proposed approach was applied to complex/realistic two and three-dimensional subsurface reservoirs and was capable of reducing the number of nonlinear iterations without compromising on the accuracy of the results. We also present an analysis of the performance of the presented acceleration technique. This analysis includes the use of different machine learning models, different dimensionless parameters, and different online learning strategies. In this study, we were able to select the most appropriate number of features (dimensionless numbers) used as inputs, and the most suitable machine learning models in terms of accuracy, prediction time and simulation results. Furthermore, we fully integrated a machine learning model into the nonlinear solver of an open-source, multiphase flow simulator (IC-FERST) and applied it to a set of challenging multiphase porous media flow problems. We were able to reduce simulation time by 37% on average, and up to 85% in the best case (only using the offline training, the online update would further improve the results). Compared to other approaches our method is simple to implement and does not require retraining the machine learning model offline when applied to other simulation domains or sets of parameters. Furthermore, not only does it learn offline, but also learns during the simulation to which it is applied. We believe that the online/adaptive learning acceleration is not limited to the multiphase porous media flow. It can be applicable to any other system for which a relaxation technique (or other tuning parameter) can be used to stabilize the nonlinear solver.

CRediT authorship contribution statement

Vinicius L.S. Silva: Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Pablo Salinas:** Writing – review & editing, Software, Methodology, Formal analysis, Conceptualization. **Claire E. Heaney:** Writing – review & editing, Supervision, Formal analysis. **Matthew D. Jackson:** Writing – review & editing, Supervision, Formal analysis, Conceptualization. **Christopher C. Pain:** Writing – review & editing, Supervision, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to acknowledge financial support from Petrobras for the first author. This work is also supported by the following EPSRC, United Kingdom grants: MUFFINS, MUltiphase Flow-induced Fluid-flexible structure InteractioN in Subsea applications (EP/P033180/1); the PREMIERE programme grant (EP/T000414/1); and SMARTRES, Smart assessment, management and optimization of urban geothermal resources (NE/X005607/1).

Table 6
Hyperparameters used for the online training of the xgboost with the boosting strategy.

Parameter	Value
Number of new boosting trees	1
Learning rate	0.01
Maximum depth	3
Subsample of instances	1.0
Subsample of columns	1.0
Buffer size	50

Table 7
Hyperparameters used for the online training of the random forest with the bagging strategy.

Paramater	Value
Number of new trees	70
Maximum depth	30
Maximum number of features	0.2
Buffer size	50

Appendix. Online training hyperparameters

For the xgboost with the boosting strategy and the random forest with the bagging strategy, we have tested different hyperparameters and the best results were achieved using the values in Tables 6 and 7, respectively.

Data availability

The data, hardware configuration, and source code used in this work are available at https://github.com/viluiz/solver.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aziz, K., 1979. Petroleum reservoir simulation. Appl. Sci. Publ. 476.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G., 2022. Deep neural networks and tabular data: A survey. IEEE Trans. Neural Networks Learn. Syst..

Breiman, L., 1999. Pasting small votes for classification in large databases and on-line. Mach. Learn. 36 (1), 85–103.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Breiman, L., et al., 1998. Arcing classifier (with discussion and a rejoinder by the author). Ann. Statist. 26 (3), 801–849.

Brooks, R., Corey, T., 1964. Hydraulic properties of porous media. Hydrol. Pap. Color. State Univ. 24, 37.

Cauwenberghs, G., Poggio, T., 2000. Incremental and decremental support vector machine learning. Adv. Neural Inf. Process. Syst. 13.

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining. pp. 785–794.

Chen, Z., Liu, B., 2018. Lifelong machine learning. Synth. Lect. Artif. Intell. Mach. Learn. 12 (3), 1–207.

Debbabi, Y., Jackson, M.D., Hampson, G.J., Fitch, P.J., Salinas, P., 2017a. Viscous crossflow in layered porous media. Transp. Porous Media 117 (2), 281–309.

Debbabi, Y., Jackson, M.D., Hampson, G.J., Salinas, P., 2017b. Capillary heterogeneity trapping and crossflow in layered porous media. Transp. Porous Media 120 (1), 183–206.

Debbabi, Y., Jackson, M.D., Hampson, G.J., Salinas, P., 2018a. Impact of the buoyancy–Viscous force balance on two-phase flow in layered porous media. Transp. Porous Media 124 (1), 263–287.

Debbabi, Y., Stern, D., Hampson, G.J., Jackson, M.D., 2018b. Use of dimensionless scaling groups to interpret reservoir simulation results. J. Pet. Sci. Eng. 163, 270–282.

Durlofsky, L.J., 1993. A triangle based mixed finite element-finite volume technique for modeling two phase flow through porous media. J. Comput. Phys. 105 (2), 252–266.

Elman, H., Silvester, D., Wathen, A., 2005. Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics, First Oxford University Press.

Freitas, M.M.d., Souza, G.d., Souto, H.P.A., 2020. A Picard-Newton approach for simulating two-phase flow in petroleum reservoirs. Int. J. Adv. Eng. Res. Sci. 7 (4), 428–457. http://dx.doi.org/10.22161/ijaers.74.53.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. System Sci. 55 (1), 119–139.

Fung, L.-K., Hiebert, A., Nghiem, L.X., et al., 1992. Reservoir simulation with a control-volume finite-element method. SPE Reserv. Eng. 7 (03), 349–357.

Géron, A., 2019. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63, 3–42.

Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T., 2017a. Adaptive random forests for evolving data stream classification. Mach. Learn. 106 (9), 1469–1495.

Gomes, J., Pavlidis, D., Salinas, P., Xie, Z., Percival, J.R., Melnikova, Y., Pain, C.C., Jackson, M.D., 2017b. A force-balanced control volume finite element method for multi-phase porous media flow modelling. Internat. J. Numer. Methods Fluids 83 (5), 431–445.

Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep Learning. Vol. 1, MIT press Cambridge.

Greenfeld, D., Galun, M., Basri, R., Yavneh, I., Kimmel, R., 2019. Learning to optimize multigrid PDE solvers. In: International Conference on Machine Learning. PMLR, pp. 2415–2423.

Grinsztajn, L., Oyallon, E., Varoquaux, G., 2022. Why do tree-based models still outperform deep learning on typical tabular data? Adv. Neural Inf. Process. Syst. 35, 507–520.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

Hoi, S.C., Sahoo, D., Lu, J., Zhao, P., 2021. Online learning: A comprehensive survey. Neurocomputing 459, 249–289.

Hoteit, H., Firoozabadi, A., 2008. Numerical modeling of two-phase flow in heterogeneous permeable media with different capillarity pressures. Adv. Water Resour. 31 (1), 56–73.

Hsieh, J.-T., Zhao, S., Eismann, S., Mirabella, L., Ermon, S., 2019. Learning neural PDE solvers with convergence guarantees. In: International Conference on Learning Representations.

Jackson, M., Percival, J., Mostaghimi, P., Tollit, B., Pavlidis, D., Pain, C., Gomes, J., Elsheikh, A.H., Salinas, P., Muggeridge, A., et al., 2015. Reservoir modeling for flow simulation by use of surfaces, adaptive unstructured meshes, and an overlapping-control-volume finite-element method. SPE Reserv. Eval. Eng. 18 (02), 115–132.

Jenny, P., Lee, S.H., Tchelepi, H.A., 2006. Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media. J. Comput. Phys. 217 (2), 627–641.

Jiang, J., Tchelepi, H.A., 2019. Nonlinear acceleration of sequential fully implicit (SFI) method for coupled flow and transport in porous media. Comput. Methods Appl. Mech. Engrg. 352, 246–275.

Kadeethum, T., O'Malley, D., Ballarin, F., Ang, I., Fuhg, J.N., Bouklas, N., Silva, V.L.S., Salinas, P., Heaney, C.E., Pain, C.C., et al., 2022. Enhancing high-fidelity nonlinear solver with reduced order model. Sci. Rep. 12 (1), 1–15.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al., 2017. Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. 114 (13), 3521–3526.

Küttler, U., Wall, W.A., 2008. Fixed-point fluid–structure interaction solvers with dynamic relaxation. Comput. Mech. 43 (1), 61–72.

Li, B., Tchelepi, H.A., 2015. Nonlinear analysis of multiphase transport in porous media in the presence of viscous, buoyancy, and capillary forces. J. Comput. Phys. 297, 104–131.

Losing, V., Hammer, B., Wersing, H., 2016. KNN classifier with self adjusting memory for heterogeneous concept drift. In: 2016 IEEE 16th International Conference on Data Mining. ICDM, IEEE, pp. 291–300.

Lott, P., Walker, H., Woodward, C., Yang, U., 2012. An accelerated Picard method for nonlinear systems related to variably saturated flow. Adv. Water Resour. 38, 92–101.

Montiel, J., Mitchell, R., Frank, E., Pfahringer, B., Abdessalem, T., Bifet, A., 2020. Adaptive XGBoost for evolving data streams. In: 2020 International Joint Conference on Neural Networks. IJCNN, IEEE, pp. 1–8.

Obeysekara, A., Salinas, P., Heaney, C.E., Kahouadji, L., Via-Estrem, L., Xiang, J., Srinil, N., Nicolle, A., Matar, O.K., Pain, C.C., 2021. Prediction of multiphase flows with sharp interfaces using anisotropic mesh optimisation. Adv. Eng. Softw. 160, 103044. http://dx.doi.org/10.1016/j.advengsoft.2021.103044.

Oladokun, E., Sheth, S., Jönsthövel, T., Neylon, K., 2020. Machine-learning informed prediction of linear solver tolerance for non-linear solution methods in numerical simulation. In: ECMOR XVII. 2020, (1), European Association of Geoscientists & Engineers, pp. 1–10.

Ortega, J.M., Rheinboldt, W.C., 1970. Iterative solution of nonlinear equations in several variables. In: Computer Science and Applied Mathematics.

Oza, N.C., Russell, S.J., 2001. Online bagging and boosting. In: International Workshop on Artificial Intelligence and Statistics. PMLR, pp. 229–236.

Pavlidis, D., Xie, Z., Percival, J.R., Gomes, J.L., Pain, C.C., Matar, O.K., 2014. Two- and three-phase horizontal slug flow simulations using an interface-capturing compositional approach. Int. J. Multiph. Flow 67, 85–91.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Polikar, R., Upda, L., Upda, S.S., Honavar, V., 2001. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Trans. Syst. Man, Cybern. Part C (Appl. Reviews) 31 (4), 497–508.

Pour, K.M., Voskov, D., Bruhn, D., 2023. Nonlinear solver based on trust region approximation for CO2 utilization and storage in subsurface reservoir. Geoenergy Sci. Eng. 225, 211698.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 2007. Numerical Recipes: The Art of Scientific Computing. Cambridge University Press.

Read, J., Bifet, A., Pfahringer, B., Holmes, G., 2012. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In: International Symposium on Intelligent Data Analysis. Springer, pp. 313–323.

Salinas, P., Pain, C., Osman, H., Jacquemyn, C., Xie, Z., Jackson, M., 2020. Vanishing artificial diffusion as a mechanism to accelerate convergence for multiphase porous media flow. Comput. Methods Appl. Mech. Engrg. 359, 112535.

Salinas, P., Pavlidis, D., Xie, Z., Adam, A., Pain, C., Jackson, M., 2017a. Improving the convergence behaviour of a fixed-point-iteration solver for multiphase flow in porous media. Internat. J. Numer. Methods Fluids 84 (8), 466–476.

Salinas, P., Pavlidis, D., Xie, Z., Jacquemyn, C., Melnikova, Y., Jackson, M.D., Pain, C.C., 2017b. Improving the robustness of the control volume finite element method with application to multiphase porous media flow. Internat. J. Numer. Methods Fluids 85 (4), 235–246.

Shwartz-Ziv, R., Armon, A., 2022. Tabular data: Deep learning is not all you need. Inf. Fusion 81, 84–90.

Silva, V.L.S., Salinas, P., Jackson, M., Pain, C., 2021. Machine learning acceleration for nonlinear solvers applied to multiphase porous media flow. Comput. Methods Appl. Mech. Engrg. 384, 113989.

Wang, H., Fan, W., Yu, P.S., Han, J., 2003. Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 226–235.

Wong, Z.Y., Kwok, F., Horne, R.N., Tchelepi, H.A., 2019. Sequential-implicit Newton method for multiphysics simulation. J. Comput. Phys. 391, 155–178.

Younis, R.M., Tchelepi, H.A., Aziz, K., 2010. Adaptively localized continuation-Newton method—nonlinear solvers that converge all the time. SPE J. 15 (02), 526–544.